2.3 PLATFORM SIGNATURE

Signature is an attribute of a platform which allows it to be detected by a sensor. The two basic types of signatures are reflective and emittive. Reflective signatures are measures of the amount of energy reflected from a platform component when it is intercepted by energy transmitted by an external source. Emittive signatures are measures of the amount of energy emanating from the platform component itself. Emittive signatures can be produced as a by-product of platform operation; e.g., the infrared (IR) energy produced by the engines of an aircraft or the acoustic signal produced by the movement of a ship. Emittive signatures can also be produced when a platform is explicitly transmitting an energy signal for some purpose; e.g., a sensor signal generated by a submarine to identify a possible target or a radio signal from a SAM commander to communicate with a subordinate. These emissions may be detectable by sensors such as warning receivers.

SWEG can simulate both types of signatures in both the electromagnetic (EM) and acoustic spectra. Reflective signatures are used only in detection calculations for an active sensor, reflecting only the energy that originated at the sensor transmitter. Emittive signatures are used only for passive sensor detection calculations. In SWEG, emittive signatures are divided into two categories, those that are the result of explicitly modeled transmitters and those that are not. For purposes of this discussion, platform emissions that are not the result of explicitly modeled transmitters in SWEG will be called collateral emissions. Reflective and collateral emittive signatures are explicitly user-defined in units of area (such as m^2 or dBsm) for reflective signatures and units of radiant intensity (watts/steradian) for collateral emittive signatures. The emittive signatures due to explicit platform transmitters in SWEG are defined by the strength of the emitted signal.

Both reflective and emittive signatures can vary with frequency, viewing angle, and polarization. SWEG allows the user to define variations due to frequency and viewing angle in as much detail as desired; i.e., there are no internal limits on the number of frequency, azimuth, or elevation intervals used. The user defines these signature variations explicitly for reflective and collateral emittive signatures, emittive signatures due to explicit transmitters vary according to the antenna pattern of the transmitter. Variation due to polarization is not modeled in SWEG.

Platform signature can change due to movement; this may be especially noticeable in the acoustic spectrum (e.g., an increase in cavitation and water flow noise as ship speed increases) and in the IR portion of the EM spectrum (e.g., an increase in the heat generated by aircraft engines as speed increases). SWEG allows the user to define signature changes due to movement for reflective and collateral emittive signatures in both the EM and acoustic spectra. These additive changes can vary with speed, frequency, and viewing angle.

Similarly, the signature of a platform component can change due to a weapon launch. SWEG allows the user to define such changes for reflective and collateral emittive signatures. These incremental signature changes are defined in logarithmic units (dB) and they may vary with weapon system and ordnance type, time since decision to launch, and viewing angle. Time is measured from the launch decision because some signature changes could occur before a round is actually fired. These changes do not specifically vary by frequency, but they may be different when viewed by different sensor types, if the user so chooses.

2.3.1 Functional Element Design Requirements

This section contains all of the requirements necessary to implement the simulation of signature in SWEG.

SWEG will provide the capability for the user to define EM and acoustic reflective signatures for any platform component type. The reflective signatures will be given in units of area and may vary in a step-wise manner for any number of frequency, azimuth, and elevation intervals of any (possibly irregular) sizes defined by the user.

a. SWEG will provide the capability for the user to define EM and acoustic collateral emittive signatures for any platform component type. The emittive signatures will be given in units of radiant intensity (watts/steradian) and may vary in a step-wise manner for any number of frequency, azimuth, and elevation intervals of any (possibly irregular) sizes defined by the user.

b. For emitted energy due to explicit platform transmitters, SWEG will treat the generated power, modulated by the antenna pattern, as the emittive signature. Thus, these emittive signatures may vary as the antenna patterns for the respective system types.

c. SWEG will simulate user-defined, movement-caused changes in reflective and collateral emittive signatures for any platform component type if so instructed. These additive changes for acoustic and EM signatures may vary in a step-wise manner for any number of frequency, azimuth, and elevation intervals of any (possibly irregular) sizes defined by the user.

d. SWEG will simulate user-defined incremental changes due to weapon launch for reflective and collateral emittive signatures for any platform component type if so instructed. These acoustic or EM changes may vary in a step-wise manner for any number of azimuth, elevation, and time intervals of any (possibly irregular) sizes defined by the user. The time intervals will be measured since the decision to launch. The changes may also vary by weapon and ordnance type, as well as by type of sensor affected.

2.3.2 Functional Element Design Approach

This section is not currently available.

2.3.3 Functional Element Software Design

This section contains a table and two software code trees which describe the software design necessary to implement the requirements and design approach outlined above. Table 2.3-1 lists most of the functions found in the code trees, and a description of each function is provided. Figure 2.3-1 describes the top level C++ functions in the code for signatures. It contains the path from main to BSRVevent, and the tree includes the primary signature-related functions called under BSRVevent. Figure 2.3-2 is the code tree for BSRVevent and its subordinate functions that only relate to signature. (Other functions in BSRVevent relate to sensors in general.)

A function’s subtree is provided within the figure only the first time that the function is called. Some functions are extensively called throughout SWEG, and the trees for these

functions are in the appendix to this document rather than within each FE description. Within this FE, the functions in that category are all the member functions in the C++ class WhereIsIt.

Not all functions shown in the figures are included in the table. The omitted entries are trivial lookup functions (single assignment statements), list-processing or memory allocation functions, or C++ class functions for construction, etc.

TABLE 2.3-1. Signature Functions Table.

Function	Description
antgeom	calculates antenna pointing and relative angle data
BaseHost::Run	runs all steps
BSRVaimpt	determines aiming point for an antenna
BSRVcalculate	determines sensor result using signal/noise calculations
BSRVevent	controls sensor physical processing
BSRVfiresig	computes increase in signature due to weapon launch
BSRVgetsigtable	gets the appropriate signature table
BSRVonechance	supervises sensor chance calculations
BSRVsignature	computes cross sectional area of a target
BSRVtgtsignal	determines total target signal power
ergazel	retrieves table entry for gain from azimuth and elevation
erggar	calculates gain and range for energy transmission
main	controls overall execution
MainInit	initiates processing and runs either the boot step or normal execution
MainParse	controls parsing of user instructions
program	controls execution of all steps except bootstrap
semant	controls semantic processing of instructions
sigmov	determines effects of movement on sensor detection
simnxt	controls event sequencing and runtime execution
simphy	controls processing of physical events
simul8	controls semantic processing of runtime instructions
srhpro	searches table for interval containing a specific value
TMemory::LLSTremove	returns a pointer to the block on the traversed list which matches the provided key
TTable::SearchInt	searches a table for a specific integer
WhereIsIt::CalcPosition	determines position for a platform given a time
WhereIsIt::CalcSpeed	determines speed for a platform given time
WhereIsIt::CalcUnitVel	determine unit velocity for a platform given a time
WhereIsIt::CalcUpVector	calculates the local up direction vector for platform
WhereIsIt::CalcVelocity	determine the local velocity vector at a specific time

```
main
    |-BaseHost::Run
        |-MainInit
            |-program
                |-MainParse
                    |-semant
                        |-simul8
                            |-simnxt
                                |-simphy
                                    |-BSRVevent
                                        |-BSRVonechance
                                            |-BSRVsignature
                                            |    |-BSRVaimpt
                                            |    |-BSRVgetsigtable
                                            |    |-BSRVgetfiresig
                                            |    |-sigmov
                                            |-BSRVcalculate
                                                 |-BSRVtgtsignal
```

FIGURE 2.3-1. Signature Top-Level Code Tree.

```
BSRVevent
    |-BSRVonechance
        |-BSRVsignature
        |    |-TMemory::Index2Ptr
        |    |-operator-
        |    |-WhereIsIt::CalcPosition
        |    |-DVector::Norm
        |    |-BSRVaimpt
        |    |    |-DVector::DVector
        |    |    |-dist
        |    |    |-DVector::Getx
        |    |    |-DVector::Gety
        |    |    |-DVector::Getz
        |    |    |-dbg_atan2
        |    |    |    \-MathExcpt::MathExcpt
        |    |    |         \-SwegExcpt::SwegExcpt
        |    |    |-WhereIsIt::CalcPosition
        |    |    |-DVector::Putz
        |    |    |-DVector::Norm
        |    |    |-FloatVector::operator=
        |    |    |    |-DVector::Getx
        |    |    |    |-DVector::Gety
        |    |    |    \-DVector::Getz
        |    |    |-operator-
        |    |    |    |-DVector::DVector
        |    |    |    |-DVector::Getx
```

FIGURE 2.3-2. Signature Code Tree.

```
|       |       |       |-DVector::Gety
|       |       |       \-DVector::Getz
|       |       |-DVector::GetHorizLength
|       |       \-FloatVector::Putz
|       |-DVector::operator+=
|       |-TMemory::Ptr2Index
|       |-WhereIsIt::CalcUnitVel
|       |-WhereIsIt::CalcUpVector
|       |-DVector::VecAng
|       |       |-operator^
|       |       |-operator*
|       |       |-DVector::DVector
|       |       |-DVector::GetLength
|       |       |-isZeroEquiv
|       |       \-dbg_atan2
|       |-TTable::SearchInt
|       |       \-TMemory::Ptr2Index
|       |-BSRVgetsigtable
|       |       |-TMemory::Index2Ptr
|       |       \-srhpro
|       |-ergazel
|       |       |-TMemory::Index2Ptr
|       |       |-srhpro
|       |       \-TMemory::Ptr2Index
|       |-BSRVfiresig
|       |       |-TMemory::Index2Ptr
|       |       |-TTable::SearchInt
|       |       |-srhpro
|       |       |-TMemory::Ptr2Index
|       |       |-dbg_pow
|       |       |       \-MathExcpt::MathExcpt
|       |       |-ergazel
|       |       \-TMemory::LLSTremove
|       |-WhereIsIt::CalcSpeed
|       |-sigmov
|       |       |-TMemory::Index2Ptr
|       |       \-srhpro
|       \-FloatVector::operator=
\-BSRVcalculate
        |-BSRVtgtsignal
                |-TMemory::Index2Ptr
                |-erggar
                |       |-antgeom
                |       |       |-DVector::DVector
                |       |       |-WhereIsIt::CalcPosition
```

FIGURE 2.3-2. Signature Code Tree. (Contd.)

```
|       |       |-DVector::operator=
|       |       |       |-FloatVector::Getx
|       |       |       |-FloatVector::Gety
|       |       |       \-FloatVector::Getz
|       |       |-DVector::DVector
|       |       |-operator-
|       |       |-DVector::Norm
|       |       |-DVector::GetHorizLength
|       |       |-DVector::operator
|       |       |-DVector::Getx
|       |       |-DVector::Getz
|       |       |-DVector::Gety
|       |       |-WhereIsIt::CalcUnitVel
|       |       |-WhereIsIt::CalcUpVector
|       |       |-operator*
|       |       |-operator+
|       |       |-operator*
|       |       |-FloatVector::Getx
|       |       |-FloatVector::Gety
|       |       \-FloatVector::Getz
|       |-DVector::VecAng
|       |       \-DVector::GetLength
|       |-TMemory::Index2Ptr
|       |-srhpro
|       |-TMemory::Ptr2Index
|       \-ergazel
|-operator-
|-WhereIsIt::CalcVelocity
|-DVector::GetLength
|-WhereIsIt::CalcPosition
|-operator^
|-operator/
|-srhpro
\-dbg_pow
```

FIGURE 2.3-2. Signature Code Tree. (Contd.)

2.3.4 Assumptions and Limitations

- All PLAYER activities can be categorized as one of six generic functions: moving, shooting, communicating, sensing, disrupting, and thinking.
- PLATFORMS leave the exercise without leaving any physical evidence.
- Signature variations due to polarization are not modeled in SWEG.

2.3.5 Known Problems or Anomalies

None.